JaxoDraw: A graphical user interface for drawing Feynman diagrams. Version 2.0 release notes.

D. Binosi

ECT, Villa Tambosi, Strada delle Tabarelle 286, I-38050 Villazzano (Trento), Italy*

J. Collins

Physics Department, Pennsylvania State University, USA

C. Kaufhold

Striekenkamp 89, 28777 Bremen, Germany

L.Theussl

Niels Bohr Institute, Copenhagen University, Blegdamsvej 17, Copenhagen 2100, Denmark

Abstract

A new version of the Feynman graph plotting tool JaxoDraw is presented. Version 2.0 is a fundamental re-write of most of the JaxoDraw core and some functionalities, in particular importing graphs, are not backward-compatible with the 1.x branch. The most prominent new features include: drawing of Bézier curves for all particle modes, on-the-fly update of edited objects, multiple undo/redo functionality, the addition of a plugin infrastructure, and a general improved memory performance. A new LaTeX style file is presented that has been written specifically on top of the original axodraw.sty to meet the needs of this this new version.

Key words: Feynman diagrams, LaTeX, Java, GUI
PACS: 01.30.Rr, 03.70.+k, 07.05.Bx

Email addresses: d.binosi@gmail.com (D. Binosi), collins@phys.psu.edu (J. Collins), jaxodraw@chka.de (C. Kaufhold), ltheussl@gmail.com (L.Theussl)

PROGRAM SUMMARY

Title of program: `JaxoDraw`
Distribution format: gzipped tar archive
Operating system:
Any Java-enabled platform, tested on Linux, Windows XP, Mac OS X
Keywords: Feynman diagrams, LaTeX, Java, GUI
Programming language used: Java
License: GPL
Catalogue identifier of previous version: ADUA
Journal Reference of previous version: Comput. Phys. Commun. 161 (2004) 76–86
Does the new version supersede the previous version?: Yes
Nature of problem:
Existing methods for drawing Feynman diagrams usually require some 'hard-coding' in one or the other programming- or scripting language. It is not very convenient and often time consuming, to generate relatively simple diagrams.
Method of solution:
A program is provided that allows for the interactive drawing of Feynman diagrams with a graphical user interface. The program is easy to learn and use, produces high quality output in several formats and runs on any operating system where a Java Runtime Environment is available.
Reasons for the new version:
A variety of new features and bug fixes.
Summary of revisions:
Major revisions since the last published user guide were versions `1.1`, `1.2` and `1.3` with several minor bug-fix releases in between.
Restrictions:
To make use of the latex export/preview functionality, a latex style file has to be installed separately. Certain operations (like internal latex compilation, Postscript preview) require the execution of external commands that might not work on untested operating systems.
Typical running time: As an interactive program, the running time depends on the complexity of the diagram to be drawn.

LONG WRITE-UP

1. Introduction

Since the first public release of `JaxoDraw-1.0` [1] in September 2003, the program has been continuously improved by the authors, partly motivated by the need of correcting obvious flaws and adding relevant new features, but above all by the overwhelming user feedback and encouragement by the community. During the last five years, `JaxoDraw` has become one of the most popular tools to draw Feynman diagrams, as witnessed by the number of citations and acknowledgements in scientific papers, as well as positive recommendations in public physics forums and reviews. The program is now included and packaged by default by various Linux distributions and has been incorporated into large-scale physics applications like the `jHepWork` [2] framework.

Version `2.0` is a major upgrade which has seen a fundamental re-write of most of the `JaxoDraw` core with respect to the `1.x` line of development. The latter had already produced some major upgrades (versions `1.1`, `1.2` and `1.3`) with several minor bug-fix releases in between.

The current document only outlines the major changes with respect to the features described in the published User Guide [1] for `JaxoDraw-1.1`. The complete User Guide for the current version is included in the program and may be consulted on the `JaxoDraw` web site, see sec. 5 for some links.

Please refer to [1] and to the present paper if using version `2.0` of the program.

1.1. Overview of main new features

The following list gives a quick overview of the most prominent new features:

- Added a plugin infrastructure. Users can write plugins for custom import/export formats that can be installed independently and will be recognized by new versions of `JaxoDraw`.
- Added Bézier curves as drawing style. Béziers can be drawn for all particle modes, including gluons and photons.
- The dimensions and positions of arrows can be adjusted.

- Added scroll bars to the drawing area. Now if an object is resized or moved beyond the current canvas size, scroll bars will appear.
- Added multiple undo/redo functionality. The user is not bound to a single undo operation, but can undo (and redo) a number of steps.
- Editing objects from the edit panel now has an immediate effect on the object so that editing operations can be previewed on the fly.

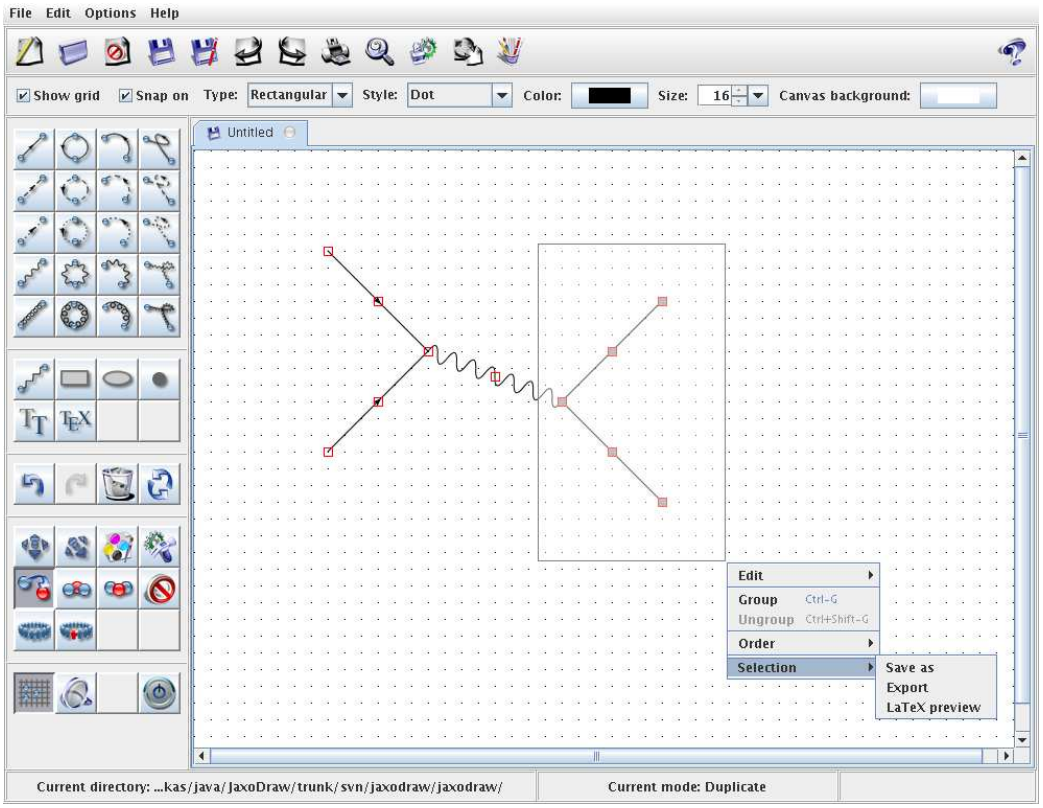

Figure 1: `JaxoDraw` in action.

Due to extensive refactoring, `JaxoDraw-2.0` *is not compatible with any earlier version of the program. In particular, xml files that were generated with earlier versions will generally not be imported correctly.*

2. Changes between v. 1.3-2 and v. 2.0

In this section we list the main changes that happened between the last released version of `JaxoDraw` and the new version `2.0`.

General.

- `JaxoDraw` now uses its own version of the axodraw style file, called `axodraw4j.sty`. This style file has to be installed separately. See App. A for a description of `axodraw4j.sty`.
- Version `2.0` has seen a number of fundamental changes in the underlying class structure of `JaxoDraw`. These changes were motivate mainly by performance and maintainability arguments, but it implies a certain backward incompatibility as graphs that where saved/exported with older versions of `JaxoDraw` will not be opened/imported correctly anymore.
- Improved memory performance which facilitates working with several graphs at the same time.
- Several key strokes have been set to more "standard" combinations. These include: Copy (`Ctrl+C`), Paste (`Ctrl+V`), Cut (`Ctrl+X`), Save (`Ctrl+S`), Save As (`Ctrl+Shift+S`), Import (`Ctrl+R`), Export (`Ctrl+Shift+R`).
- The "Copy" mode has been renamed to "Duplicate" to avoid confusion with the usual clipboard "Copy" action.
- SVG support has been removed from the core program, it is now available as a separate plugin.
- The preferences are not stored in a file `.Jaxorc` anymore, the standard Java Preferences API is used instead. See Sec. (4.1) for details.
- The User Guide is not bundled in the jar archive anymore, but created at run-time and installed into a local directory. See Sec. (4.2) for details.
- Log records are now written to a rotating sequence of files in a default log directory. See Sec. (4.3) for details.

- The drawing of photons and gluons has been adjusted to use the same algorithms as the postscript code in the `axodraw4j` style file. Different output formats are entirely consistent now, however some strange effects happen for special configurations, e.g. gluon loops with very small radii.

New features.

- Added a plugin infrastructure. Users can write plugins for custom import/export formats that can be installed independently and will be recognized by new versions of `JaxoDraw`. See Secs. (4.4,4.5) for details.
- Added Bézier curves as drawing style. Béziers can be drawn for all particle modes, including gluons and photons.
- Photon and gluon arcs are now painted during resizing.
- Make the symmetry of a photon arc configurable.
- Editing objects from the edit panel now has an immediate effect on the object so that editing operations can be previewed on the fly.
- Added the new feature that a double click with the right mouse button on the canvas brings up the edit panel for the nearest object.
- The faint box method for selecting objects via a right click and drag action on the Canvas has been improved. On button release, a menu will be presented with a choice of operations applicable to the current selection of enclosed objects. In particular the selected objects can be previewed, exported and/or saved.
- Added scroll bars to the canvas. Now if an object is resized or moved beyond the current canvas size, scroll bars will appear.
- Added multiple undo/redo functionality. The user is not bound to a single undo operation, but can undo (and redo) a number of steps determined by the `Undo depth` preference (currently limited to 10^4 moves).
- Groups can be rescaled by dragging, like all other objects.

- Objects with fill color can now be optionally unfilled (only drawing their boundary).
- A grid bar has been added to the main panel which shows at a glance all the grid settings of the current tab.
- The grid can now be customized. In particular several grid styles are now available: `dot`, `cross`, `line` and `honeycomb` (the latter being available only when the grid type is set to `hexagonal`).
- Holding down the mouse button after a center click on the canvas remove all handles that might be cluttering a graph in Edit mode.
- Arrows can be customized and arbitrarily positioned. The parameters that can be controlled are the arrow `length`, `width` and `inset`. This works also with LaTeX/LaTeX`->EPS` export, thanks to the new sty file `axodraw4j`.
- Different exporting formats have now different options.
- Warning system when exporting with unsupported options.
- Color space can be set by the user. The available options are `Axodraw` (coinciding with the usual `colordvi` color space) and `complete` (representing the complete RGB space). When working with the complete color space and exporting to LaTeX/LaTeX`->EPS`, color conversion will be applied.
- Edited objects are automatically brought in the forefront, for better visualization of the editing operations. After the finalization of the editing operations, they will go back to the original ordering.
- Edit panels are brought up in positions that avoid covering the edited object as much as possible.
- A lot of new preferences can be now set via the Preferences panel.
- The `Copy` action from different Tabs has been streamlined, through the use of the faint box method.
- Multiple files can be specified on the command line and will be opened in multiple tabs.

- The command line option `-nosplash` can now be used in order not to show the splash window on startup.
- The command line option `--convert` can now be used to convert a number of `JaxoDraw` xml files to `axodraw4j` tex files (and vice versa) without the need of bringing up the user interface.

Bug fixes.

- Fixed the bug where file operations (Open, Save, Import, ...) did not work on Windows if the path to the file contained whitespaces.
- Fixed the bug that gluon loops did not close in latex output. This was a bug in `axodraw.sty` that is fixed in the new `axodraw4j.sty`.
- Fixed the bug that double line separation and line width was defined inconsistently which led to different results in latex and postscript output.
- Prevent various user input dialogs from going into the background and blocking the main window.
- Fixed the bug that made `JaxoDraw` hang if `axodraw.sty` was not installed or not found.
- The hexagonal grid strategy has been changed to make the grid uniform, not subject to rounding, which could make points drop from the grid if objects were moved.
- A large number of minor bug fixes that are detailed in the `CHANGES` document of the source distribution.

3. New features since v. 1.1

This section only lists the new features that were added at every major release since version `1.1`, which is the version that was described in our first published description of the program [1]. Each of the releases below also incorporated a number of bug fixes that can be tracked from the corresponding release notes. In addition, a few point releases were also made (`1.0-1`, `1.3-1`, `1.3-2`) with only minor bug fixes.

3.1. New features in v. 1.3

- Make the Mac OS X README file available from the Help menu (Mac only).
- New vertex type diamond.
- Implemented LaTeX text rotation (using the pstricks package) and rotation of Postscript texts.
- Added default return mode.
- New export/preview formats SVG, JPG and PNG.
- Added a dynamic zoom.
- Rewrote the export dialog for key-friendlyness: tab key toggles between items, space selects, escape cancels. In the export formats combobox you can choose an entry by pressing its first character or go up and down with the arrow keys.

3.2. New features in v. 1.2

- Added a hexagonal grid. Each tab can have its own grid type and size.
- Introduced a 'WatchFile' mode to avoid opening new windows for each preview.
- Introduced radio buttons in the vertex menus to indicate the currently active Vertex mode.
- PSText now can display curly brackets like in LaTeX: \{ and \}.
- Several Mac OS X specific enhancements, eg. menu key short cuts, key short cuts for middle and right mouse button, a preference for the latex and dvips path, which allows internal latex compilation, etc.
- Arcs and triangular vertices are now three-point objects: they are drawn with a click for each point.
- The Preferences panel has been restructured for a clearer layout.
- Many more fonts are now available in PSText mode because we do not filter out fonts anymore that cannot display greek characters.

4. Guide to new features

This section describes in more detail some of the most prominent new features of JaxoDraw-2.0 with respect to the last released version.

4.1. Local configuration directory and preferences

The preferences are not stored in a file .Jaxorc (in the user's home directory $USER_HOME/) anymore, the standard Java Preferences API is used instead. A folder $USER_HOME/.jaxodraw/$VERSION/ (in the following called $JAXO_DIR) is used to store all program-specific information, currently there are sub-directories for log files, plugins and the User Guide.

4.2. User guide

The User Guide is not bundled in the distributed program (jar archive) anymore, but created at run-time and installed into $JAXO_DIR/usrGuide/. It can therefore be opened by any custom browser and be viewed locally and independently of JaxoDraw.

4.3. Logging

Log records are now written to a rotating sequence of files in the default log directory $JAXO_DIR/log/. The logging level for the written log records is always kept at DEBUG, only the logging level for the console output can be configured (e.g. with the --debug or --quiet command-line options).

4.4. Plugins

In version 2.0 a plugin architecture was added to JaxoDraw. Plugins are software components that may optionally be added to the program at runtime, ie without the need of changing or re-compiling the main program. This makes it easy to use optional features, in particuar export to uncommon formats or import of other custom file formats, while keeping the size of the main program at a minimum. In fact, the original main purpose of the plugin architecture was to draw some functionality out of the JaxoDraw core.

Plugins are installed (and un-installed) using the Plugin Manager panel which is accessible from the Options menu. Once a plugin is installed, JaxoDraw will automatically recognize it at start-up and the corresponding functionality will be available for the current session and subsequent sessions until the plugin is uninstalled. Plugins are installed in $JAXO_DIR/plugins/.

A list of available plugins is maintained on the JaxoDraw web site, currently there are plugins available for export to PDF (Portable Document Format) and SVG (Scalable Vector Graphics) format.

4.5. Writing custom import/export plugins

The most interesting consequence of the plugin architecture is that it allows anybody to write custom plugins that may be loaded by `JaxoDraw` and used by anybody without re-compiling the main program.

As an illustration, imagine we have a Feynman diagram coded in some custom input file (eg created by another program), and we would like to import this diagram into `JaxoDraw` so that it can be edited interactively. All we have to do is write a Java class that extends `JaxoImportPlugin` and implement all the required abstract methods, the most important in this case being `importGraph`:

```
public class MyImportJaxoPlugin extends JaxoImportPlugin
{
    protected JaxoGraph importGraph(InputStream inputStream)
        throws JaxoPluginExecutionException
    {
        // return a graph read from an InputStream
    }
}
```

A ready-compiled plugin can then be installed by the `JaxoDraw` Plugin Manager without changing anything in the main program. In particular, we can publish the plugin so other people can install and use it as well, without having to download and install a new version of `JaxoDraw`!

For more detailed instructions and further information on writing plugins please refer to the `JaxoDraw` web site.

5. Additional information

The `JaxoDraw` home page is at `http://jaxodraw.sourceforge.net/`. It contains up-to-date information about the program, in particular links to our mailing lists[1] and bug-tracking system[2].

Acknowledgements

We would like to thank all the people on the `JaxoDraw` mailing list for their help and feedback during the testing phase.

[1] `http://jaxodraw.sourceforge.net/mail-lists.html`

[2] `http://jaxodraw.sourceforge.net/issue-tracking.html`

A. New features of `axodraw4j`

`JaxoDraw-2.0` uses a new LaTeX style file called `axodraw4j.sty` (i.e. '`axodraw` for `JaxoDraw`') for its LaTeX exports which is based on J. Vermaseren's original `axodraw.sty` [3]. The name has been changed to avoid any backward compatibility issues with old documents that use the original `axodraw`.

We describe here, from a user's perspective, how `axodraw4j.sty` differs from the original version. The reader unfamiliar with `axodraw` should therefore first refer to the documentation for the original version[3].

A.1. Main changes from axodraw to axodraw4j

- Lines (solid, dashed, photon, and gluon) can now be made double, with an adjustable separation.
- The dimensions and positions of arrows can be adjusted.
- Lines and dashed lines can be made from Bézier curves.
- Since there are now many more possibilities to specify a line, optional arguments to the main line drawing commands can be used to specify them in a keyword style.
- A new macro named `\Arc` is introduced for arcs and dashed arcs.
- For consistency the `\GlueArc` macro is renamed to `\GluonArc`, with the old macro retained as a synonym.
- Some bugs are corrected. The most notable one is that `axodraw4j` now works correctly with `revtex` and `revtex4`.
- The behavior of arcs is changed when the specified opening angle is outside the natural range.
- The macros originally specified as `\B2Text`, `\G2Text`, and `\C2Text` are now named `\BTwoText`, `\GTwoText`, and `\CTwoText`.

[3] `http://www.ctan.org/get/graphics/axodraw/axodraw.pdf`

A.2. Commands

The `axodraw4j.sty` file has been kept largely backward compatible with the original `axodraw`, i.e. almost all commands that were available in `axodraw` are also implemented in `axodraw4j` (the exception being the renaming of commands like `\BTwoText`, etc.). The few additional commands and the enhanced old commands are documented below. For a description of the remaining commands, please refer to the original documentation of `axodraw` [3].

A typical use of the macros in a document is as follows:

```
\documentclass{article}
\usepackage{axodraw4j}
\begin{document}
 \begin{picture}(162,39) (0,0)
   \Line[arrow,arrowlength=5,arrowwidth=2](0,19)(48,19)
   \Arc[arrow,arrowlength=5,arrowwidth=2](80,43)(40,143,36)
   \Line(112,19)(160,19)
   \GluonArc(80,-5)(40,37,143){3.5}{6}
 \end{picture}%
\end{document}
```

The main macros for line drawing are `\Arc`, `\Bezier`, `\Gluon`, `\GluonArc`, `\Line`, `\Photon`, and `\PhotonArc`. In each of the following descriptions of the macros, the part enclosed in square brackets, "`[`*options*`]`", is an optional argument, with the options being specified by keywords, as explained later.

- `\Arc[`*options*`](`x`,`y`)(`r`,`ϕ_1`,`ϕ_2`)`

  Draws a circular arc. The center of the arc is (x,y), the radius is r, and the starting and ending angles are ϕ_1 and ϕ_2 (in degrees). By default, the arc is an anticlockwise single solid line without an arrow.

  Supported option groups are: `arrow`, `clock`, `dash`, `double`.

- `\Bezier[`*options*`](`x_1`,`y_1`)(`x_2`,`y_2`)(`x_3`,`y_3`)(`x_4`,`y_4`)`

  Draws a Bézier line with control points (x_1,y_1), (x_2,y_2), (x_3,y_3), and (x_4,y_4).

  Supported option groups are: `arrow`, `dash`, `double`.

- \Gluon[*options*](*x*1,*y*1)(*x*2,*y*2){*amplitude*}{*windings*}

  Draws a gluon from (x_1,y_1) to (x_2,y_2). The width of the gluon is twice the 'amplitude' parameter, and the number of windings is the 'windings' parameter, which is rounded down to an integer. The side at which the windings lie is determined by the order of the two coordinates. Also a negative amplitude changes this side.

  Supported option groups are: `double`.

- \GluonArc[*options*](x,y)(r,ϕ_1,ϕ_2){*amplitude*}{*windings*}

  Draws a gluon on a circular arc. The center of the arc is (x,y), the radius is r, and the starting and ending angles are ϕ_1 and ϕ_2 (in degrees). By default, the arc is anticlockwise.

  The width of the gluon is twice the 'amplitude' parameter, and the number of windings is the 'windings' parameter, which is rounded down to an integer. Whether the curls are inside or outside depends on the sign of the amplitude. When it is positive the curls are on the inside.

  Supported option groups are: `clock`, `double`.

- \Line[*options*](*x*1,*y*1)(*x*2,*y*2)

  Draws a line from (x_1,y_1) to (x_2,y_2). By default the line is a solid single line without an arrow.

  Supported option groups are: `arrow`, `dash`, `double`.

- \Photon[*options*](*x*1,*y*1)(*x*2,*y*2){*amplitude*}{*wiggles*}

  Draws a photon from (x_1,y_1) to (x_2,y_2). The width of the gluon is twice the 'amplitude' parameter, and the number of windings is the 'windings' parameter. The number of windings is rounded down to an integer or half integer. Whether the first wiggle starts up or down depends on the sign of the amplitude. If the amplitude (rounded) is a half integer, the photon is symmetric.

  Supported option groups are: `double`.

- \PhotonArc[*options*](x,y)(r,ϕ_1,ϕ_2){*amplitude*}{*wiggles*}

  Draws a photon on a circular arc. The center of the arc is (x,y), the radius is r, and the starting and ending angles are ϕ_1 and ϕ_2 (in degrees). The width of the gluon is twice the 'amplitude' parameter, and

the number of windings is the 'windings' parameter, which is rounded down to an integer. By default, the arc is anticlockwise.

The sign of the amplitude determines whether the photon starts going outside (positive) or starts going inside (negative). If the photon is to reach both endpoints from the outside the number of wiggles should be an integer plus 0.5.

Supported option groups are: `clock`, `double`.

A.3. Options

Options are organized in logical groups for different kinds of property, and are specified by keyword-value pairs, as in

```
\Line[arrow=true,arrowlength=5,arrowwidth=2](0,19)(48,19)
```

For a boolean keyword, the keyword alone is equivalent to the true value. Before options are processed, from left to right, the values are set to initial default values. For the boolean options, the initial values are all false.

Each command only implements a subset of the keywords. The possible keywords, organized by groups, are

`dash` Boolean option specifying that the line is dashed.

`dashsize` The size of the dashes. Defaults to 3.

`double` Boolean option specifying that the line is doubled.

`linesep` The separation of a double line. It is the distance between the center of the two component single-lines. Defaults to 2.

`clock` Boolean option specifying that an arc is drawn clockwise from the starting angle instead of anticlockwise.

`arrow` Boolean option specifying that an arrow is drawn on the line.

If neither of the arrow dimensions (length or width) is specified, the (half-)width defaults to $1.2 * (2 +$ `linewidth`$)$ for a single line, and to $1.2 * (2 + 0.7 *$ `linesep` $+$ `linewidth`$)$ for a double line, and the length defaults to $2.5 *$ `arrow-half-width`. If only one of the width and length is specified, the other is determined by the same aspect ratio as for a default arrow.

`arrowpos` The position of the arrow on an object (line / arc / loop / bezier). It is given as a fraction between 0 and 1. Defaults to 0.5 (i.e., the arrow is at the middle of the object).

`arrowscale` If neither an explicit arrow length nor an explicit arrow width is given for a line, then the value given with this keyword specifies the linear scale of the arrow relative to the default size of the arrow.

`arrowlength` The length of the arrow.

`arrowidth` The half-width of the arrow.

`arrowinset` The inset of the tail of an arrow. It is specified as a fraction of the arrowlength, a number between 0 and 1. Defaults to 0.2.

`flip` Boolean option specifying that the arrow is reversed (flipped) relative to the direction of the line, i.e., that it points from the end towards the start of the line.

Macros are available for setting certain parameters without the need to specify them in individual line commands. In each of the descriptions below, `num` represents a number

`\SetArrowScale{`*num*`}` Sets the scale of an arrow relative to the default. This is used on a line with an arrow when the dimensions are not otherwise specified. Its value is initialized to 1.

`\SetArrowInset{`*num*`}` Sets the inset of the tail of an arrow relative to its length. This is used on a line with an arrow when the arrowinset is not explicitly specified. Its value is initialized to 0.2.

`\SetArrowAspect{`*num*`}` Sets the default aspect ratio of arrows; this is the ratio of the length of an arrow to its *full* width. This is used in arrows when no dimensions are specified or when only one of the length and width are specified. Its value is initialized to 1.25.

`\SetArrowPosition{`*num*`}` Sets the default position of arrows along a line: 0 is at the start, 1 is at the end, 0.5 is at the center. This is used on a line with an arrow when the arrowpos option is not used. Its value is initialized to 0.5.

References

[1] D. Binosi and L. Theussl, *JaxoDraw: A graphical user interface for drawing Feynman diagrams*, Comput. Phys. Commun. **161**, 76 (2004) [arXiv:hep-ph/0309015].

[2] S. Chekanov, *HEP data analysis using jHepWork and Java*, `arXiv:0809.0840` [cs.CE].

[3] J. A. M. Vermaseren, *Axodraw*, Comput. Phys. Commun. **83**, 45 (1994).